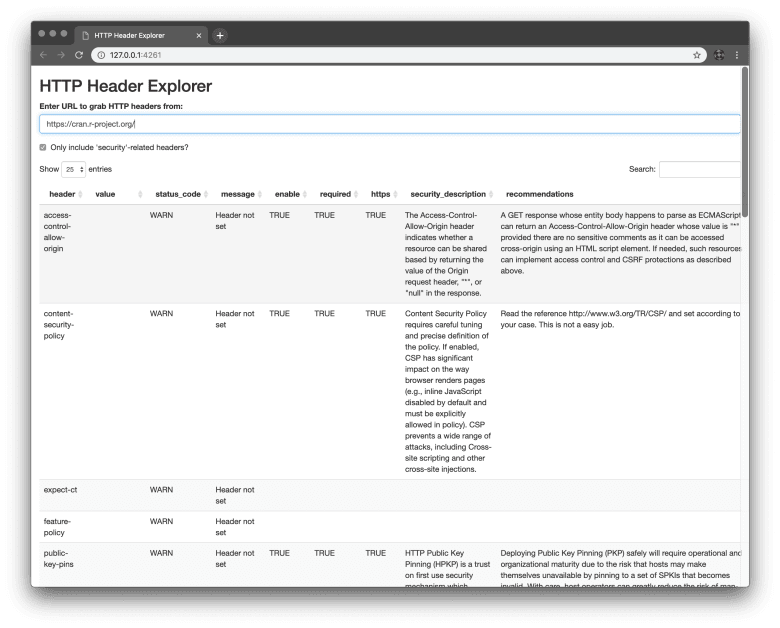
In the event you wanted to dig deeper I [threw together](https://git.rud.is/hrbrmstr/hdrs) a small package that will let you grab HTTP headers from a given URL and take a look at them. The README has examples for most things but we’ll go through a bit of them here as well.

For those that just want to play, you can do:

install.packages("hdrs", repos = "https://cinc.rud.is/")

hdrs::explore\_app()

[](https://i0.wp.com/rud.is/b/wp-content/uploads/2019/03/shiny-headers.png?ssl=1)

and use the diminutive Shiny app to explore a site’s security headers or look at all the headers they return. (*Oh, yeah…if you read the previous post then looked at the above screenshot you’ll notice how completely useless IP blocking is to determined ~~attackers~~ individuals.*)

NOTE: There are binaries for macOS and Windows at my CINC repo for hdrs so you’ll be getting those if you use the above method. Use type='source' on that call or use various remotes package functions to install the source package (after reading it b/c you really shouldn’t trust any package, ever) from:

* [SourceHut](https://git.sr.ht/~hrbrmstr/hdrs)
* [GitLab](https://gitlab.com/hrbrmstr/hdrs)
* if you must [GitHub](https://github.com/hrbrmstr/hdrs)

**Moving Ahead**

Let’s use the command-line to poke at my newfound most favorite site to use in security-related examples:

library(hdrs)

assess\_security\_headers("https://cran.r-project.org/") %>%

dplyr::select(-url)

## # A tibble: 13 x 4

## header value status\_code message

## \*

## 1 access-control-allow-origin NA WARN Header not set

## 2 content-security-policy NA WARN Header not set

## 3 expect-ct NA WARN Header not set

## 4 feature-policy NA WARN Header not set

## 5 public-key-pins NA WARN Header not set

## 6 referrer-policy NA WARN Header not set

## 7 server Apache/2.4.10 (Debian) NOTE Server header found

## 8 strict-transport-security NA WARN Header not set

## 9 x-content-type-options NA WARN Header not set

## 10 x-frame-options NA WARN Header not set

## 11 x-permitted-cross-domain-policies NA WARN Header not set

## 12 x-powered-by NA WARN Header not set

## 13 x-xss-protection NA WARN Header not set

Ouch. Not exactly a great result (so, perhaps it matters little how poorly maintained the downstream mirrors are after all, or maybe it’s *perfectly fine* to run a [five year old web server](http://mail-archives.apache.org/mod_mbox/httpd-announce/201407.mbox/%3C650BABAF-9B03-4EEB-94EC-D6DD833C248F@apache.org%3E) with some fun [vulns](https://httpd.apache.org/security/vulnerabilities_24.html)).

Anyway…

The assess\_security\_headers() function looks at 13 modern “security-oriented” HTTP headers, performs a very light efficacy assessment and returns the results.

* access-control-allow-origin
* content-security-policy
* expect-ct
* feature-policy
* server
* public-key-pins
* referrer-policy
* strict-transport-security
* x-content-type-options
* x-frame-options
* x-permitted-cross-domain-policies
* x-powered-by
* x-xss-protection

Since you likely do not have every HTTP header’s name, potential values, suggested values, and overall purpose memorized, you can also pass in include\_ref = TRUE to the function to get more information with decent textual descriptions like you saw in the screenshot (the Shiny app omits many fields).

The full reference is available in a data element:

data("http\_headers")

dplyr::glimpse(http\_headers)

## Observations: 184

## Variables: 14

## $ header\_field\_name "A-IM", "Accept", "Accept-Additions", "Accept-Charset", "Accept-Datetime", "Accept-Encoding…

## $ type\_1 "Permanent", "Permanent", "Permanent", "Permanent", "Permanent", "Permanent", "Permanent", …

## $ protocol "http", "http", "http", "http", "http", "http", "http", "http", "http", "http", "http", "ht…

## $ status "", "standard", "", "standard", "informational", "standard", "", "standard", "", "standard"…

## $ reference "https://tools.ietf.org/html/rfc3229#section-10.5.3", "https://tools.ietf.org/html/rfc7231#…

## $ type\_2 "Request", "Request", "Request", "Request", "Request", "Request", "Request", "Request", "Re…

## $ enable FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, FALSE, TRUE, FALSE, FAL…

## $ required NA, NA, NA, NA, NA, NA, NA, NA, TRUE, TRUE, NA, TRUE, NA, NA, NA, TRUE, NA, NA, NA, NA, NA,…

## $ https NA, NA, NA, NA, NA, NA, NA, NA, TRUE, TRUE, NA, TRUE, NA, NA, NA, TRUE, NA, NA, NA, NA, NA,…

## $ security\_description "", "", "", "", "", "", "", "", "Sometimes an HTTP intermediary might try to detect viruses…

## $ security\_reference "", "", "", "", "", "", "", "", "https://tools.ietf.org/html/rfc5789#section-5", "https://t…

## $ recommendations "", "", "", "", "", "", "", "", "Antivirus software scans for viruses or worms.", "Servers …

## $ cwe "", "", "", "", "", "", "", "", "CWE-509: Replicating Malicious Code (Virus or Worm)", "CWE…

## $ cwe\_url "\r", "\r", "\r", "\r", "\r", "\r", "\r", "\r", "https://cwe.mitre.org/data/definitions/509…

There will eventually be a lovely vignette with well-formatted sections that include the above information so you can reference it at your leisure (it’s *great* bedtime reading).

The http\_headers object is fully documented but here’s what those fields mean:

* header\_field\_name: header field
* type\_1: Permanent (in a standard); Provisional (experimental); Personal (unofficial)
* protocol: should always be http for now but may be different (e.g. quic)
* status: blank == unknown; otherwise the value describes the status well
* reference: where to look for more info
* type\_2: Request (should only be found in requests); Response (should only be found in responses); Request/Response found in either; Reserved (not in use yet)
* enable: should you have this enabled
* required: Is this header required
* https: HTTPS-specific header?
* security\_description: Information on the header
* security\_reference: Extra external reference information on the header
* recommendations: Recommended setting(s)
* cwe: Associated Common Weakness Enumeration (CWE) identifier
* cwe\_url: Associated CWE URL

**Even Moar Headers**

HTTP servers can spit out tons of headers and we can catch’em all with hdrs::explain\_headers(). That function grabs the headers, merges in the full metadata from http\_headers and returns a big ol’ data frame. We’ll only pull out the security reference URL for this last example:

explain\_headers("https://community.rstudio.com/") %>%

dplyr::select(header, value, security\_reference)

## # A tibble: 18 x 3

## header value security\_reference

##

## 1 cache-control no-cache, no-store https://tools.ietf.org/html/rfc7234#…

## 2 connection keep-alive ""

## 3 content-encoding gzip https://en.wikipedia.org/wiki/BREACH…

## 4 content-security-po… base-uri 'none'; object-src 'none'; script-src 'unsafe-eval'… https://www.owasp.org/index.php/List…

## 5 content-type text/html; charset=utf-8 https://tools.ietf.org/html/rfc7231#…

## 6 date Tue, 05 Mar 2019 20:40:31 GMT ""

## 7 referrer-policy strict-origin-when-cross-origin NA

## 8 server nginx https://tools.ietf.org/html/rfc7231#…

## 9 strict-transport-se… max-age=31536000 https://tools.ietf.org/html/rfc6797

## 10 vary Accept-Encoding ""

## 11 x-content-type-opti… nosniff https://www.owasp.org/index.php/List…

## 12 x-discourse-route list/latest NA

## 13 x-download-options noopen NA

## 14 x-frame-options SAMEORIGIN https://tools.ietf.org/html/rfc7034

## 15 x-permitted-cross-d… none NA

## 16 x-request-id 12322c6e-b47e-4960-b384-32138097886c NA

## 17 x-runtime 0.106664 NA

## 18 x-xss-protection 1; mode=block https://www.owasp.org/index.php/List…

**FIN**

Have some fun and poke at some headers. Perhaps even use this to do a survey of key web sites in your field of work/study and see how well they rate. As usual, post PRs & issues at your fav social coding site.